Claims

1. A method of creating a logical device performing polynomial division, comprising:

(a) using a hardware description language to build code directly describing synthesizable logic for performing the polynomial division; and

(b) implementing the logic on a target device, wherein the code receives as inputs a parameter identifying a polynomial and a parameter identifying a number of data bits for which the polynomial division is performed.

2. A method according to claim 1 wherein the logical device is used in one or more of the following:

(a) a cyclic redundancy checker;

(b) a cyclic redundancy calculator;

(c) a scrambler;

(d) an error correction device; and

(e) a component of an error correction scheme.

3. A method according to claim 1 wherein the target device includes at least one of:

(a) a field programmable gate array;

(b) an application specific integrated circuit; and

(c) any other suitable logic device on which the logic may be implemented.

4. A method of creating a logical device performing polynomial division for a given n-degree polynomial including calculating a next n-term remainder for a data unit having d terms, the method including:

(a) creating code using a high level description language directly describing synthesizable logic for performing the polynomial division including:

(i) automatically extracting a subset of data terms for calculating each of the next remainder terms; and

(ii) calculating the next remainder by performing a logical XOR operation on the subset of data terms with a subset of relevant remainder terms calculated for a preceding data unit;

and

(b) implementing the logic on a target device.

5. A method according to claim 4 wherein the logical XOR operation includes a pipelined XOR operation with a pre-definable number of pipeline stages, the pipelined XOR operation operating on the subset of data terms.

6. A method according to claim 5 wherein the number of pipeline stages is greater than or equal to 1.

7. A method according to claim 4 wherein the subset of data terms is extracted by:

(a) identifying data terms in an incoming data unit which are required for calculating a respective next remainder term; and

(b) performing a first logical AND operation on the identified data terms with the incoming data unit;

wherein performing the first logical AND operation eliminates data terms not required for calculating the next remainder.

8. A method according to claim 7 wherein identifying the incoming data terms required for calculating a respective next remainder term includes creating a data-enable vector with d terms by, for each of the data terms:

(a) resetting all terms in the data unit;

(b) asserting a current data term;

(c) determining a logic equation for calculating the next remainder; and

(d) if the asserted data term is present in the logic equation, asserting a corresponding term of the data-enable vector;

wherein the identified data terms used in the first logical AND operation are corresponding terms of the data-enable vector.

9. A method according to claim 4 wherein the subset of remainder terms is automatically extracted by, for each of the remainder terms:

(i) identifying remainder terms calculated for a preceding data unit which are required to calculate the next remainder, and

(ii) performing a second logical AND on the identified remainder terms with the remainder terms calculated for the preceding data unit.

10. A method according to claim 9 wherein identifying remainder terms calculated for a preceding data unit which are required to calculate the next remainder includes creating a remainder term-enable vector with n terms by, for each of the remainder terms:

(a) resetting all remainder terms;

(b) asserting a current remainder term;

(c) determining a logic equation for calculating the next remainder; and

(d) if the asserted remainder term is present in the logic equation, asserting a corresponding term of the remainder term-enable vector;

wherein the identified remainder terms used in the second logical AND operation are corresponding terms of the remainder term-enable vector.

11. A method according to claim 4 wherein the logical device is used in one or more of the following:

(a) a cyclic redundancy checker;

(b) a cyclic redundancy calculator;

(c) a scrambler;

(d) an error correction device; and

(e) a component of an error correction scheme.

12. A method according to claim 4 wherein the target device includes at least one of:

(a) a field programmable gate array;

(b) an application specific integrated circuit; and

(c) any other suitable logic device on which the logic may be implemented.

13. A computer program product residing on a programmable medium containing hardware description language code directly describing synthesizable logic suitable for implementation on a target device conveying a programmed method of performing polynomial division on units of data each having d terms and using a polynomial of degree n to calculate a next remainder having n terms, the programmed method comprising:

(i) automatically extracting a subset of data terms for calculating each of the next remainder terms;

and

(ii) calculating the next remainder by performing a logical XOR operation on the subset of data terms with a subset of relevant remainder terms calculated for a preceding data unit.

14. A computer program product according to claim 13 wherein the logical XOR operation includes a pipelined logical XOR operation with a pre-definable number of pipeline stages, the pipelined logical XOR operation operating on the subset of data terms.

15. A computer program product according to claim 14 wherein the number of pipelined stages is greater than or equal to 1.

16. A computer program product according to claim 13 wherein the programmed method further comprises extracting the subset of data terms by:

(a) identifying data terms in an incoming data unit which are required for calculating a respective next remainder term; and

(b) performing a first logical AND operation on the identified data terms with the incoming data unit

17. A computer program product according to claim 16 wherein the programmed method further comprises creating a data-enable vector with d terms to identify the incoming data terms required for calculating a respective next remainder term, the data-enable vector being created by, for each of the data terms:

(a) resetting all terms in the data unit;

(b) asserting a current data term;

(c) determining a logic equation for calculating the next remainder; and

(d) if the asserted data term is present in the logic equation, asserting a corresponding term of the data-enable vector;

wherein the identified data terms used in the first logical AND operation are corresponding terms of the data-enable vector.

18. A computer program product according to claim 13 wherein the programmed method further comprises identifying the subset of relevant remainder terms calculated for the previous data unit automatically by, for each of the remainder terms:

(i) identifying remainder terms calculated for the preceding data unit which are required to calculate the next remainder, and

(ii) performing a second logical AND on the identified remainder terms with the remainder terms calculated for the preceding data unit.

19. A computer program product according to claim 18 wherein the programmed method further comprises identifying the remainder terms calculated for the preceding data unit which are required to calculate the next remainder by creating a remainder term-enable vector having n terms, the remainder term-enable vector being created by, for each of the remainder terms:

(a) resetting all remainder terms;

(b) asserting a current remainder term;

(c) determining a logic equation for calculating the next remainder; and

(d) if the asserted remainder term is present in the logic equation, asserting a corresponding term of the remainder term-enable vector;

wherein the identified remainder terms used in the second logical AND operation are corresponding terms of the remainder term-enable vector.